
GA based robust blind digital watermarking

Víctor Álvarez · José Andrés Armario ·
María Dolores Frau · Félix Gudiel ·
María Belén Güemes · Elena Martín ·
Amparo Osuna

Abstract A genetic algorithm based robust blind digital watermarking scheme is presented. Starting from a binary image (the original watermark), a genetic algorithm is performed searching for a permutation of this image which is as uncorrelated as possible to the original watermark. The output of the GA is used as our final watermark, so that both security and robustness in the watermarking process is improved. Now, the original cover image is partitioned into non-overlapped square blocks (depending on the size of the watermark image). Then a (possibly extended) Hadamard transform is applied to these blocks, so that one bit information from the watermark image is embedded in each block by modifying the relationship of two coefficients in the transformed matrices. The watermarked image is finally obtained by simply performing the inverse (extended) Hadamard transform on the modified matrices. The experimental results show that our scheme keeps invisibility, security and robustness more likely than other proposals in the literature, thanks to the GA pretreatment.

Keywords watermarking · genetic algorithm

V. Álvarez, J.A. Armario, M.D. Frau, F. Gudiel, E. Martín, A. Osuna
Dept. Matemática Aplicada 1, ETSII, Avda. Reina Mercedes s/n, 41012, Sevilla, Spain
Tel.: +34-95455-2797, -4386, -4389, -6225, -2798, -2798
Fax: +34-954557878
E-mail: {valvarez,armario,mdfrau,gudiel,emartin,aosuna}@us.es

M.B. Güemes
Dept. Algebra, Fac. Matemáticas, Avda. Reina Mercedes s/n, 41012, Sevilla, Spain
Tel.: +34-954556969
Fax: +34-954556938
E-mail: bguemes@us.es

1 Introduction

Digital watermarking concerns those methods about how to hide a special mark into digital multimedia data to solve the problems of legal ownership, integrity and authenticity of the original data [2].

The techniques proposed so far can be grouped into two different approaches, depending on whether the watermark is embedded into the least significant bits (spatial domain approach, [9]) or it is embedded attending to the perceptually most significant frequency components of the container image (frequency domain approach, [1]). Usually one tends to apply techniques of the second type, since spatial domain approaches have relatively low information hiding capacity and, what is more important, can be easily erased by lossy image compression.

Most of frequency domain approaches use discrete wavelet transform (DWT), discrete Fourier transform (DFT) and discrete cosine transform (DCT). Very recently fast Hadamard transform (FHT) has arisen as a promising alternative (see [5] and [12] for instance). Interested readers in *Hadamard matrices* (square matrices consisting in pairwise orthogonal rows) are referred to [6].

No matter the processing speed is, watermarking is usually required to muster three conditions: *security*, *imperceptibility* and *robustness*.

Security is concerned with embedding a watermark into a piece of content at an untrusted user device without compromising the security of the watermark key, the watermark or the original (see [7] for instance).

Perceptibility measures whether perceptible artifacts on the watermarked image are introduced, that is, if the presence of the watermark in the final image is noticeable. This magnitude is measured in terms of the *Peak Signal to Noise Ratio*, or *PSNR* in brief. It is most easily defined via the *mean squared error* (*MSE*), so that for images with maximum possible pixel value *range* (i.e. 255 or 1 depending on whether byte or real storing method is adopted), *PSNR* is calculated as:

$$PSNR = 10 \log_{10} \frac{range^2}{MSE}. \quad (1)$$

Here, for two $m \times n$ monochrome images $K = (k_{i,j})$ and $L = (l_{i,j})$ (where one of the images is considered a noisy approximation of the other), *MSE* is defined as follows:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (k_{i,j} - l_{i,j})^2.$$

The robustness of a watermark depends on whether it fails to be detected after unintentional or even malicious transformations (see [10] for details). It is usually measured in terms of the *Normalized Correlation* (*NC*) between the extracted watermark image $EW = (ew_{i,j})$ (presumably modified) and the

original watermark $W = (w_{i,j})$,

$$NC_{W,EW} = \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} w_{i,j} ew_{i,j}}{\sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} w_{i,j}^2} \sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} ew_{i,j}^2}}. \quad (2)$$

Given a watermarking scheme, it may be straightforwardly improved in terms of security, imperceptibility and robustness by simply introducing some pretreatment to the watermark image in order to destroy space relativity (see [11] or [12] for instance).

Taking the work in [12] as starting point, in this paper we describe an improved blind (that is, the original cover image is not needed for extracting the watermark) watermarking scheme, with the following advantages:

- There is no dependence on the sizes of the watermark and cover images (in [12] it is forced to be 1/8).
- The trade-off between imperceptibility and robustness is measured in terms of a parameter b . The greater b is, the nearer NC is to 1, the smaller $PSNR$ is. Accordingly, the smaller b is, the smaller NC is, the greater $PSNR$ is. Some explanations (beyond simple computational evidence!) will be given in order to justify the optimal value for b , depending on the way in which the image is being stored (real or byte representation).
- Robustness, security and imperceptibility of the scheme are significantly improved thanks to a pretreatment of the watermark image. We have designed a genetic algorithm (in the sequel, GA in brief), looking for a permutation of the original watermark which is as uncorrelated as possible to it. This GA, equipped with a specific crossover operator specially designed for the occasion, beats usual GA equipped with classical crossover operators concerning permutations problems (such as *order 1*, *partially mapped* and *cycle* crossovers, or *edge recombination*).

We organize the paper as follows. Section 1 is devoted to introduce the problem of watermarking, and our proposal. The GA looking for permuted images of the watermark with low correlation is described in Section 2. Section 3 is devoted to explain the watermarking scheme. Some executions and examples are showed in Section 4. We include a last section for conclusions and comments.

2 GA for uncorrelated permuted images

Given an image Im , we want to find a permuted image PIm of Im , so that their normalized correlation $NC(Im, PIm)$ is as less as possible. The key problem here is establishing a method for looking for permutations of the watermark as uncorrelated as possible. Since one cannot afford to perform an

exhaustive search in the full set of permutations, we are dealing with a problem worth of some kind of heuristic solution.

Permutation encoded problems are often used to represent scheduling problems and classic combinatorial optimization problems, such as the Traveling Salesman Problem, Bin Packing or Job Scheduling, to name some. Here, the goal consists in (or can be solved by) arranging some objects in a certain order.

These optimization problems can seldom be solved by using exhaustive searches, due to the large size of the search space. Some kind of heuristic is required instead. For instance, GAs.

In what follows, we assume that the reader is familiar with the general framework of GAs, and their usual elements and characteristics. If necessary, [3] (and the references there included) is a good place to get a general overview of the subject.

The GAs which deal with permutations are known as *ordering GAs* or simply *order-based GAs*.

Although GAs are widely recognized as powerful and widely applicable optimization methods for string encoded problems, there is no standard GA for manipulating ordered-list representations. However, several crossover operators and mutations have been suggested in the literature, including *Order Crossover (OX)*, *Partially Mapped Crossover (PMX)*, *Cycle Crossover (CX)*, *Edge Recombination (ER)*, *Insert Mutation (InsM)*, *Swap Mutation (SwM)*, *Inversion Mutation (InvM)* or *Scramble Mutation (ScM)*, for instance. The interested reader is referred to [3] and [8] for details and further bibliography.

Here we use a *Steady-State GA*, in the sense that three offspring (two coming from crossover, and one more coming from mutation) are generated per generation, which will replace the worst adapted individuals at the moment. This way, we use also *elitism*, since we always keep the fittest solution so far. Additionally, we use a “no duplicates” policy, in the sense that identical individuals are not allowed to occur in the same generation.

In order to select two individuals for reproduction purposes, we use *rank-based* selection, by means of *linear ranking*. More concretely, assume that the size of the population is μ . Then sort the population in terms of fitness, so that fittest has rank μ and worst rank 1. Now fix a factor $1 \leq s \leq 2$ (we use $s = 1.5$ in the sequel). In this circumstances, the probability that the i^{th} individual is selected for reproduction is given by

$$P_i = \frac{2-s}{\mu} + \frac{2(i-1)(s-1)}{\mu(\mu-1)} \quad (3)$$

Depending on whether the factor s is closer to 1, fitness is accordingly relativized for choosing the individual.

Our population consists of 20 individuals, and every run is limited to 50 generations. The fitness function consists in the normalized correlation (2) between the permuted image and the original watermark, so that the lesser NC is, the fitter an individual is.

Although defining crossover operators for permutation encoding is a difficult task, nevertheless, we have designed a new crossover operator (which we denote simply by X), attending to the characteristics of our problem.

Since our watermark images are binary (just black ($=0$) and white ($=1$) pixels) $m \times m$ matrices, we can easily encode them as m^2 -length binary vectors. Furthermore, we can just save the positions in which 1 (analogously, 0) entries are displayed. Assume that the watermark image consists of k white pixels. Then the set of its permuted images is uniquely determined by the set of k -subsets of $\{1, \dots, m^2\}$.

Given two such different k -subsets S_1 and S_2 , we select proportionally positions in S_1 and S_2 attending to their fitness. Assume that S_1 has better fitness than S_2 . Fix randomly a real number $0.5 \leq r \leq 1$. Then we will get a $\lfloor k \cdot r \rfloor$ -subset S of S_1 , and join this subset with a random $k - \lfloor k \cdot r \rfloor$ -subset of $S_2 - S$.

A priori, there is no evidence of which among the crossover operations cited above is more suitable for our purposes.

The comparison in Table 2 suggests that our proposed crossover operation X is more successful than traditional ordering crossovers OX , PMX , CX and ER .

Due to the binary character of the watermarks, we have chosen two images with significantly different density of white pixels (the *FAMA* logo of the University of Seville and the common *Stop* driving signal, see Table 2).



Table 1 *FAMA* and *STOP* watermarks.

For each of these images, we have performed 10 runs for each crossover and mutation operators, each of which consisted of 50 generations. The size of every population is fixed in 20 individuals. In the table below, NC_i denotes the best NC value at generation i , Av_i denotes the average of NC_i values along the 10 runs, and $iter$ denotes the average of the generation in which fittest NC_{50} was found along the 10 runs. The optimal values for NC found so far are written in bold. They have been obtained performing a GA consisting of our proposed crossover operator X and the Inverse Mutation operator.

Notice that there is no interest in timing considerations here, since this is a preprocessing step in our watermarking scheme. Anyway, it is noticeable that every run is performed in just a few seconds.

From Table 2 above we conclude that our GA looking for a minimally correlated permuted image is successful, in the sense that normalized correlations of random permuted images (forming the initial populations) are always

		<i>FAMA</i>					<i>STOP</i>				
		NC_0	Av_0	NC_{50}	Av_{50}	<i>iter.</i>	NC_0	Av_0	NC_{50}	Av_{50}	<i>iter.</i>
OX	<i>InsM</i>	0.8048	0.8076	0.8021	0.8037	36	0.1872	0.1968	0.1694	0.1803	38
	<i>SwM</i>	0.8042	0.8069	0.8009	0.8030	28	0.1794	0.1935	0.1750	0.1819	39
	<i>InvM</i>	0.8051	0.8077	0.8021	0.8040	35	0.1839	0.1957	0.1727	0.1804	37
	<i>ScM</i>	0.8024	0.8068	0.8009	0.8034	29	0.1850	0.1982	0.1694	0.1794	41
PMX	<i>InsM</i>	0.8057	0.8078	0.8000	0.8017	43	0.1850	0.1956	0.1627	0.1706	44
	<i>SwM</i>	0.8057	0.8075	0.7994	0.8023	43	0.1928	0.1986	0.1694	0.1761	41
	<i>InvM</i>	0.8057	0.8071	0.8000	0.8017	44	0.1895	0.1974	0.1594	0.1682	42
	<i>ScM</i>	0.8045	0.8077	0.7976	0.8016	44	0.1672	0.1921	0.1549	0.1668	44
CX	<i>InsM</i>	0.8045	0.8080	0.8036	0.8052	30	0.1884	0.1967	0.1817	0.1877	29
	<i>SwM</i>	0.8045	0.8072	0.8042	0.8055	16	0.1906	0.19810	0.1806	0.1898	29
	<i>InvM</i>	0.8045	0.8073	0.8036	0.8048	23	0.1828	0.1959	0.1750	0.1822	36
	<i>ScM</i>	0.8051	0.80786	0.8030	0.8047	21	0.1895	0.1975	0.1761	0.1857	33
ER	<i>InsM</i>	0.8039	0.8074	0.8039	0.8054	30	0.1906	0.1991	0.1806	0.1848	38
	<i>SwM</i>	0.8075	0.8082	0.8057	0.8072	21	0.1794	0.1955	0.1794	0.1898	19
	<i>InvM</i>	0.8054	0.8075	0.8036	0.8053	32	0.1850	0.1936	0.1783	0.1853	16
	<i>ScM</i>	0.8063	0.8081	0.8033	0.8055	36	0.1861	0.1966	0.1806	0.1845	33
X	<i>InsM</i>	0.8048	0.8069	0.799	0.8018	43	0.1806	0.1947	0.1560	0.1623	46
	<i>SwM</i>	0.8045	0.8081	0.8000	0.8025	40	0.1872	0.1981	0.1505	0.1591	48
	<i>InvM</i>	0.8057	0.8079	0.7952	0.7993	39	0.1806	0.191	0.1159	0.1343	45
	<i>ScM</i>	0.8036	0.8072	0.7994	0.8007	45	0.1884	0.1986	0.1449	0.1581	43

Table 2 GA applied to *FAMA* and *STOP*.

greater than the local optimum found by the GA, no matter the crossover operator has been selected. Moreover, the crossover *X* seems to provide fitter permuted images than usual ordering crossovers.

In the following section we describe a watermarking scheme. We claim that pretreatment of the original watermark (so that a minimally correlated permuted image is obtained and used instead), improves the watermarking scheme in an obvious way, not only from the security point of view (no matter one knows the extracting procedure, the extracted watermark will be meaningless), but sometimes (depending on the concrete image, compare Tables 4 and 6 below) also from the point of view of invisibility, without loose of robustness.

3 The watermarking scheme

Grayscale digital images may be stored both in real (the real values of the pixels moving from 0=black to 1=white) or byte (the integer values of the pixels moving from 0=black to 255=white) encoding. Nevertheless, usually byte encoding is preferred, since not every computational system can support working with real encoding (and it requires a discretization step). We will work here with byte encoding, unless stated otherwise.

Let A be the original cover grayscale digital image, encoded as a $n \times n$ matrix with integer entries in $\{0, \dots, 255\}$.

Let W be the binary watermark, encoded as a $m \times m$ matrix with 0,1 entries.

3.1 Watermarking embedding

The embedding procedure may be detailed as follows:

- Find a normalized Hadamard matrix H of size $4t$ closest to $\lfloor \frac{n}{m} \rfloor$.
- Partition the original image A into non-overlapped blocks of size $\lfloor \frac{n}{m} \rfloor$. Consider the sub-blocks of size $4t \times 4t$ naturally embedded, which we denote by A_i , $1 \leq i \leq m^2$.
- Apply the *extended* Hadamard Transform to A_i , in order to obtain

$$B_i = \frac{H A_i H^T}{4t} \quad (4)$$

- Select two entries b_1 and b_2 in B_i in the same row (or column), say $b_1 = B_i(3, 3)$ and $b_2 = B_i(3, 5)$ for instance. Depending on whether the corresponding pixel i in W is 0 or 1, force that $b_2 > b_1$ or $b_2 < b_1$ accordingly. To this end, fix a value b , and take $d = \frac{|b_1 - b_2|}{2}$. Then set:
 - If $i = 0$ and $b_2 \leq b_1$ then actualize $b_1^* = b_1 - d - b$, $b_2^* = b_2 + d + b$.
 - If $i = 1$ and $b_2 \geq b_1$ then actualize $b_1^* = b_1 + d + b$, $b_2^* = b_2 - d - b$.
- The watermarked block A_i^* is obtained by the inverse transform of (4),

$$A_i^* = \frac{H^T B_i^* H}{4t} \quad (5)$$

At this point, we would like to make two major comments:

1. Taking a deeper insight in the Hadamard transform (4) one deduces that a change Δ in b_i translates into a change about $\lceil \frac{\Delta}{2t} \rceil$ in $A_i^* = \frac{H^T B_i^* H}{4t}$. Hence, in order to get noticeable byte changes, we should take $b \approx t$. In the case of real encoded images, b may be chosen arbitrarily small (at the risk of decreasing the normalized correlation of the extracted watermark).
2. Although the *Hadamard Conjecture* about the existence of these matrices in every order $4t$ remains open, there are well known families of Hadamard matrices filling an infinite amount of sizes $4t$ (see [6] for details, and [4] for a computer aided generation of Hadamard matrices).

3.2 Watermarking extraction

The extraction procedure is just the inverse procedure of embedding.

Let A_i^* be the i^{th} -block of the watermarked image. In order to recover the pixel i of the watermark one must simply proceed as follows:

- By (4), form $B_i^* = \frac{HA_i^*H^T}{4t}$.
- Let b_1 and b_2 be the entries used in the embedding procedure. If $b_2 > b_1$ set $i = 0$, and $i = 1$ otherwise.

4 Experimental results

We have used 5 different 512×512 cover images (see Table 3), and two different 64×64 watermarks (see Table 1). We have fixed H to be the 8×8 Sylvester Hadamard matrix,

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 \end{pmatrix}$$

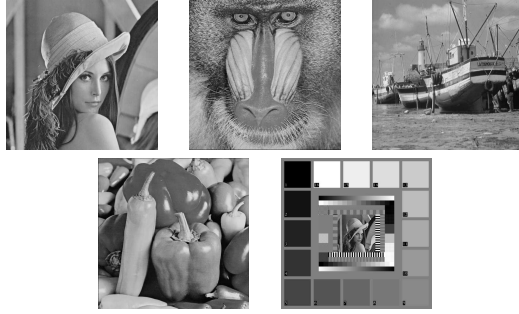






















Table 3 Cover images: Lena, baboon, boats, peppers, testlena.

Table 4 shows the $PSNR$ and NC values obtained for different values of the parameter b .

Table 5 shows that the proposed watermarking scheme is robust under different attacks, such as jpeg compression (with quality factors 80% and 90%), Gaussian noise (of mean 0 and variance 0.001) and salt-and-pepper noise (of density 0.01).

		Lena			Baboon			Boats		
<i>b</i>		<i>PSNR</i>	<i>NC</i>	<i>W*</i>	<i>PSNR</i>	<i>NC</i>	<i>W*</i>	<i>PSNR</i>	<i>NC</i>	<i>W*</i>
	2.01	47.1796	1		40.3194	1		46.4154	1	
	2	47.2432	1		40.34	1		46.4493	1	
	1.99	47.3031	0.9976		40.3604	0.9989		46.4827	0.9982	
	2.01	47.256	1		40.1841	1		46.4054	1	
	2	47.2988	1		40.2034	1		46.431	1	
	1.99	47.3414	0.9586		40.2238	0.9676		46.457	0.9696	









































		<i>Peppers</i>			<i>Testlena</i>		
<i>b</i>	<i>PSNR</i>	<i>NC</i>	<i>W*</i>	<i>PSNR</i>	<i>NC</i>	<i>W*</i>	
	2.01	47.256	1		38.0736	1	
	2	48.5626	1		38.1818	0.9711	
	1.99	48.6202	0.9982		38.3227	0.9191	
	2.01	48.2392	1		42.7214	1	
	2	48.3059	1		43.0388	0.6448	
	1.99	48.3734	0.9480		43.3333	0.4993	

Table 4 *PSNR* and *NC* in terms of *b*.

		<i>Lena</i>			<i>Baboon</i>			<i>Boats</i>		
<i>Noise</i>		<i>PSNR</i>	<i>NC</i>	<i>W*</i>	<i>PSNR</i>	<i>NC</i>	<i>W*</i>	<i>PSNR</i>	<i>NC</i>	<i>W*</i>
	<i>jpg90%</i>	39.9389	0.8758		35.3971	0.9057		38.4282	0.8722	
	<i>jpg80%</i>	38.0837	0.7747		31.9305	0.8313		36.0909	0.7859	
	<i>Gauss.</i>	47.0694	0.9969		40.3029	0.9978		46.304	0.9971	
	<i>S.&P.</i>	25.4416	0.9139		25.47	0.9238		25.5055	0.9179	
	<i>jpg90%</i>	39.9704	0.6477		35.3456	0.7294		38.4404	0.6636	
	<i>jpg80%</i>	38.0843	0.4713		31.9249	0.5663		36.0861	0.4869	
	<i>Gauss.</i>	47.1223	0.9729		40.1736	0.9884		46.2867	0.9846	
	<i>S.&P.</i>	25.4419	0.7591		25.4655	0.7781		25.5054	0.7620	



















		<i>Peppers</i>			<i>Testlena</i>		
<i>Noise</i>		<i>PSNR</i>	<i>NC</i>	<i>W*</i>	<i>PSNR</i>	<i>NC</i>	<i>W*</i>
	<i>jpg90%</i>	38.4336	0.8739		37.5428	0.9698	
	<i>jpg80%</i>	36.5783	0.7764		36.9454	0.8912	
	<i>Gauss.</i>	48.3526	0.9962		38.0697	0.9941	
	<i>S.&P.</i>	25.3601	0.9140		24.9004	0.9152	
	<i>jpg90%</i>	38.4118	0.6660		41.3027	0.7895	
	<i>jpg80%</i>	36.5486	0.4955		39.9323	0.4621	
	<i>Gauss.</i>	48.1107	0.9793		42.7086	0.9650	
	<i>S.&P.</i>	25.3589	0.7548		25.0392	0.7564	

Table 5 *PSNR* and *NC* under noises.





		<i>Lena</i>		<i>Baboon</i>		<i>Boats</i>		<i>Peppers</i>		<i>Testlena</i>	
<i>W</i>	<i>PW</i>	<i>PSNR</i>	<i>NC</i>	<i>PSNR</i>	<i>NC</i>	<i>PSNR</i>	<i>NC</i>	<i>PSNR</i>	<i>NC</i>	<i>PSNR</i>	<i>NC</i>
		47.5041	1	40.2723	1	46.4686	1	48.7436	1	43.5181	1
		47.2524	1	40.4052	1	46.2693	1	48.1666	1	43.5181	1

Table 6 *PSNR* and *NC* using GA-watermarks.

Although the proposed watermarking scheme works fine without any need of pretreatment of the watermark, we want to emphasize that permuting the initial watermark ensures that security and imperceptibility (and even robustness to a somewhat lesser degree) are enhanced.

Table 6 below shows computational evidence of this fact, when we substitute the original watermark *W* by the permuted images *PW* provided by the GA described in the previous section (those corresponding to the bold entries in Table 2).

5 Conclusions

In this paper we have described a new blind watermarking scheme. We have shown that pretreatment of the original watermark so that a minimally correlated permuted image is obtained and used instead, improves the watermarking scheme not only from the security point of view, but also from the point of view of imperceptibility, without loose of robustness (see Table 6).

Although the problem of finding a minimally correlated permuted image from the given watermark is hard, we have designed an order-based Steady-State GA which successfully solve the problem. It includes a proper crossover operator, specifically developed attending to the particular features of our problem. This crossover operator has been shown to beat classical order crossover operators experimentally (see Table 2).

Acknowledgements All authors are partially supported by FEDER funds via the research projects FQM-016 and P07-FQM-02980 from JJAA and MTM2008-06578 from MICINN (Spain).

References

1. I.J. Cox, J. Kilian, F.T. Leighton, T. Shannon, Secure spread spectrum watermarking for multimedia, *IEEE Trans. Image Processing* 8 (1997) 1673–1687.
2. I.J. Cox, M.L. Miller, J.A. Bloom, *Digital watermarking*, Academic Press, New York, 2006.

3. A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer, Natural Computing Series, 2003.
4. V.K. Gupta, R. Parsad, A. Dhaudapani, <http://www.iasri.res.in/webhadamard/webhadamard.htm>
5. A.T.S. Ho, J. Shen, S.H. Tan, A.C. Kot, Digital image-in-image watermarking for copyright protection of satellite images using the fast Hadamard transform, in: Proceedings of the 24th IEEE Int. Geoscience and Remote Sensing Symposium, Toronto, Canada, 2002, pp. 3311-3313.
6. K.J. Horadam, Hadamard matrices and their applications, Princeton University Press, Princeton, 2007.
7. A. Lemma, S. Katzenbeisser, M. Celik, M. van der Veen, Secure Watermark Embedding Through Partial Encryption, in: Proceedings IWDW2006, Eds. Y.Q. Shi and B. Jeon, LNCS 4283, Springer Verlag, Berlin Heidelberg, 2006, pp. 433-445.
8. P.W. Poon, J.N. Carter, Genetic algorithm crossover operators for ordering applications, Computers Ops. Res. 22 (1) (1995) 135-147.
9. R.G. van Schyndel, A.Z. Tirkel, C.F. Osborne, A digital watermark, in: Proceedings IEEE Int. Conf. Image Processing 2, 1994, pp. 86-90.
10. S. Voloshynovskiy, F. Deguillaume, O. Koval, T. Pun, Robust watermarking with channel state estimation, Signal Processing, special issue on: Security of Data Hiding Technologies, Eds. S. Voloshynovskiy, T. Pun, J. Fridrich, F. Pérez-González, N. Memon, 83, 10, 2003.
11. D. Zhang, J. Xu, H. Li, H. Li, A novel image watermarking algorithm with fast processing speed, in: Proceedings ICIECS 2009, IEEE Press, Wuhan, China, 2009.
12. Y. Zhang, Z.M. Lu, D.N. Zhao, A blind image watermarking scheme using fast Had